Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 4

з дисципліни «Методи оптимізації та планування експерименту»

на тему «Проведення трьохфакторного експерименту з використанням рівняння регресії з урахуванням ефекту взаємодії»

Варіант 110

ВИКОНАЛА:

студентка ІІ курсу ФІОТ

групи ІВ-71

Молчанова В.С.

ПЕРЕВІРИВ:

Асистент

Регіда П. Г.

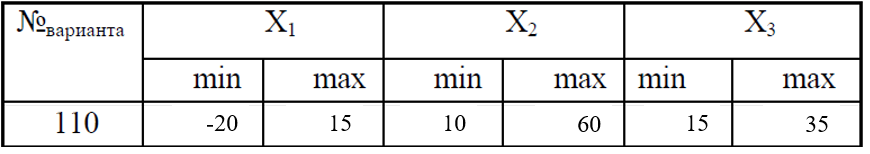
Київ – 2019

**Мета:**

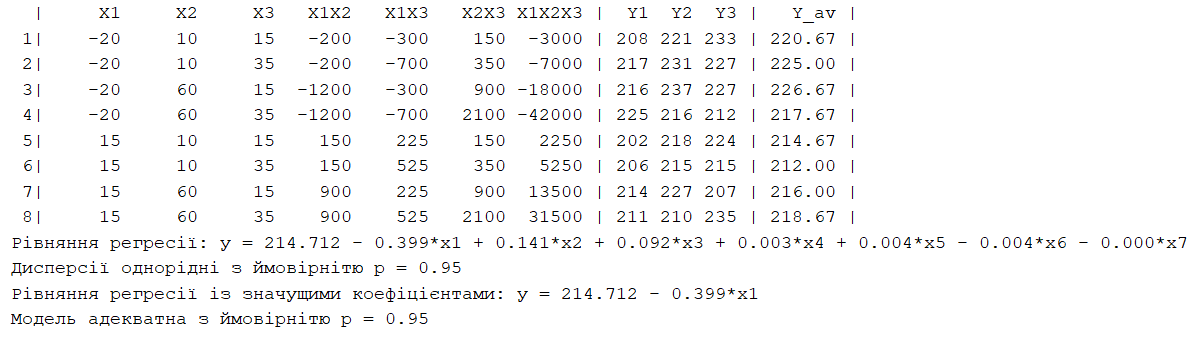
провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

**Хід роботи:**

**Варіант завдання:**



**Результати підготовки та виконання роботи:**



**Код програми:**

**import** math  
**import** numpy **as** np  
**import** random  
**import** crit  
k\_simple = 3  
k\_comb = 7  
N\_simple = 4  
N\_comb = 8  
cur\_m = 3  
p = 0.95  
  
x = [[-20, 15],  
 [10, 60],  
 [15, 35]]  
  
x\_av\_max = sum([row[1] **for** row **in** x])/3  
x\_av\_min = sum([row[0] **for** row **in** x])/3  
  
y\_max = round(200+x\_av\_max)  
y\_min = round(200+x\_av\_min)  
  
x\_norm\_simple = [[1, -1, -1, -1],  
 [1, -1, -1, 1],  
 [1, -1, 1, 1],  
 [1, 1, 1, 1]]  
x\_norm\_t\_simple = np.transpose(x\_norm\_simple)  
x\_norm\_comb = [[1, -1, -1, -1],  
 [1, -1, -1, 1],  
 [1, -1, 1, -1],  
 [1, -1, 1, 1],  
 [1, 1, -1, -1],  
 [1, 1, -1, 1],  
 [1, 1, 1, -1],  
 [1, 1, 1, 1]]  
  
x\_nat\_simple = []  
**for** i **in** range(N\_simple):  
 x\_nat\_simple.append([1])  
 **for** j **in** range(1, k\_simple+1):  
 **if** x\_norm\_simple[i][j] < 0:  
 x\_nat\_simple[i].append(x[j - 1][0])  
 **else**:  
 x\_nat\_simple[i].append(x[j - 1][1])  
  
x\_nat\_t\_simple = list(np.transpose(x\_nat\_simple))  
x\_nat\_t\_simple.pop(0)  
  
x\_nat\_comb = []  
**for** i **in** range(N\_comb):  
 x\_nat\_comb.append([1])  
 **for** j **in** range(1, k\_simple+1):  
 **if** x\_norm\_comb[i][j] < 0:  
 x\_nat\_comb[i].append(x[j - 1][0])  
 **else**:  
 x\_nat\_comb[i].append(x[j - 1][1])  
  
**for** i **in** range(N\_comb):  
 x\_norm\_comb[i].append(x\_norm\_comb[i][1] \* x\_norm\_comb[i][2])  
 x\_norm\_comb[i].append(x\_norm\_comb[i][1] \* x\_norm\_comb[i][3])  
 x\_norm\_comb[i].append(x\_norm\_comb[i][2] \* x\_norm\_comb[i][3])  
 x\_norm\_comb[i].append(x\_norm\_comb[i][1] \* x\_norm\_comb[i][2] \* x\_norm\_comb[i][3])  
  
 x\_nat\_comb[i].append(x\_nat\_comb[i][1] \* x\_nat\_comb[i][2])  
 x\_nat\_comb[i].append(x\_nat\_comb[i][1] \* x\_nat\_comb[i][3])  
 x\_nat\_comb[i].append(x\_nat\_comb[i][2] \* x\_nat\_comb[i][3])  
 x\_nat\_comb[i].append(x\_nat\_comb[i][1] \* x\_nat\_comb[i][2] \* x\_nat\_comb[i][3])  
  
x\_norm\_t\_comb = np.transpose(x\_norm\_comb)  
x\_nat\_t\_comb = list(np.transpose(x\_nat\_comb))  
x\_nat\_t\_comb.pop(0)  
  
**while** cur\_m < 6:  
 x\_nat = x\_nat\_simple  
 x\_nat\_t = x\_nat\_t\_simple  
 x\_norm = x\_norm\_simple  
 x\_norm\_t = x\_norm\_t\_simple  
 N = N\_simple  
 k = k\_simple  
 y = [[random.randint(y\_min, y\_max) **for** j **in** range(cur\_m)] **for** i **in** range(N)]  
 y\_av = [sum(row) / cur\_m **for** row **in** y]  
  
 mx = [sum(x\_nat\_t[i]) / N **for** i **in** range(k)]  
 my = sum(y\_av) / N  
 a = []  
 **for** i **in** range(k):  
 a.append([])  
 **for** j **in** range(k):  
 a[i].append(sum([a \* b **for** (a, b) **in** zip(x\_nat\_t[i], x\_nat\_t[j])]) / N)  
 a0 = [sum([a \* b **for** a, b **in** zip(x\_nat\_t[i], y\_av)]) / N **for** i **in** range(k)]  
 left = []  
 left.append(mx.copy())  
 left[0].insert(0, 1)  
 **for** i **in** range(k):  
 left.append([mx[i]])  
 **for** j **in** range(k):  
 left[i + 1].append(a[j][i])  
 right = a0.copy()  
 right.insert(0, my)  
 b = np.linalg.solve(left, right)  
  
 **if not** crit.Criteria.cohren(cur\_m, N, y\_av, y):  
 cur\_m += 1  
 **continue** t\_main = crit.Criteria.student(cur\_m, N, y\_av, y, x\_norm\_t, k)  
 b\_main = [b[i] \* t\_main[i] **for** i **in** range(k + 1)]  
  
 **if** crit.Criteria.fisher(cur\_m, N, y\_av, y, t\_main, b\_main, x\_nat, k):  
 print\_results(x\_nat, b, t\_main)  
 **break** *# With interactions* x\_nat = x\_nat\_comb  
 x\_nat\_t = x\_nat\_t\_comb  
 x\_norm = x\_norm\_comb  
 x\_norm\_t = x\_norm\_t\_comb  
 N = N\_comb  
 k = k\_comb  
 y = [[random.randint(y\_min, y\_max) **for** j **in** range(cur\_m)] **for** i **in** range(N)]  
 y\_av = [sum(row) / cur\_m **for** row **in** y]  
  
 mx = [sum(x\_nat\_t[i]) / N **for** i **in** range(k)]  
 my = sum(y\_av) / N  
 a = []  
 **for** i **in** range(k):  
 a.append([])  
 **for** j **in** range(k):  
 a[i].append(sum([a \* b **for** (a, b) **in** zip(x\_nat\_t[i], x\_nat\_t[j])]) / N)  
 a0 = [sum([a \* b **for** a, b **in** zip(x\_nat\_t[i], y\_av)]) / N **for** i **in** range(k)]  
 left = []  
 left.append(mx.copy())  
 left[0].insert(0, 1)  
 **for** i **in** range(k):  
 left.append([mx[i]])  
 **for** j **in** range(k):  
 left[i + 1].append(a[j][i])  
 right = a0.copy()  
 right.insert(0, my)  
 b = np.linalg.solve(left, right)  
  
 **if not** crit.Criteria.cohren(cur\_m, N, y\_av, y):  
 cur\_m += 1  
 **continue** t\_main = crit.Criteria.student(cur\_m, N, y\_av, y, x\_norm\_t, k)  
 b\_main = [b[i] \* t\_main[i] **for** i **in** range(k + 1)]  
  
 **if** crit.Criteria.fisher(cur\_m, N, y\_av, y, t\_main, b\_main, x\_nat, k):  
 print\_comb\_results(x\_nat, b, t\_main)  
 **break**

**Висновок:**

Експериментальні значення функцій відгуку збігаються з середніми, отже коефіцієнти рівнянь регресії знайдені правильно.